CS 330: Applied Database Systems

The Last Lecture

(Some slides courtesy of Gun Sirer)

Some Remarks

- Final on May 19
- Sample exam questions on the webpage early next week (including X** questions)

Today's Lecture

- Peer-to-peer:
  - Napster and Gnutella
  - PEPPER

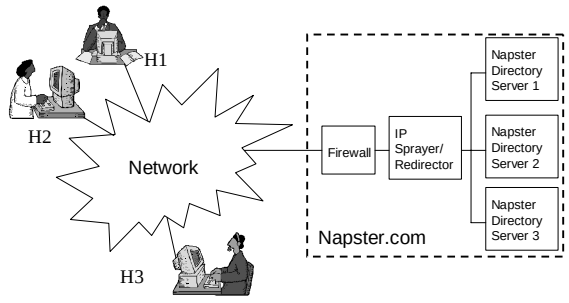
Napster

- Flat filesystem
  - Single-level filesystem with no hierarchy
  - Can have multiple files with the same name
- All storage is done at the edges
  - Each host computer exports a set of files that reside locally on that host.
  - The host is registered with a centralized directory; uses keepalives to show that it is still connected
  - A centralized directory is notified of the filenames that are exported by that host
- Simple, centralized directory

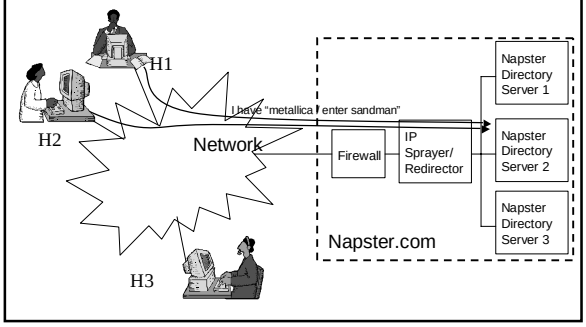
Napster Directory

- File lookup in Napster
  - Client queries directory server for filenames matching a pattern
  - Directory server picks 100 files that match the pattern, sends them to the client
  - Client pings each, computes round trip time to each host, displays results
  - User then transfers file directly from the closest host
- File transfers are peer-to-peer, with no involvement of anyone other than the two edge hosts

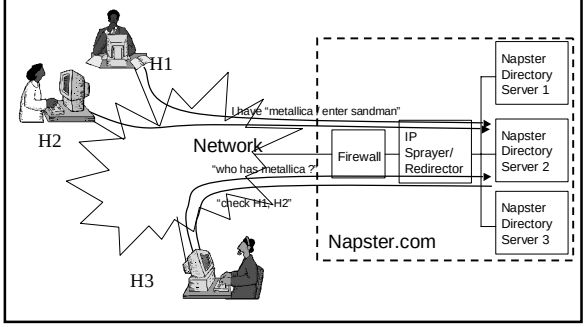
Napster Architecture



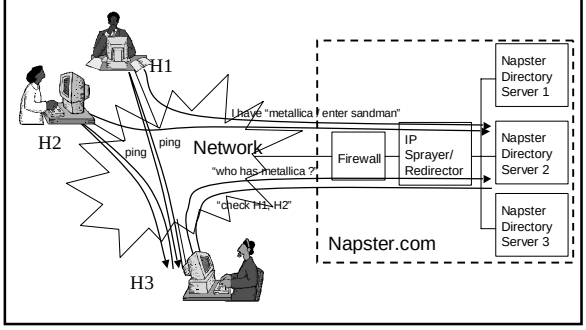
Napster Protocol



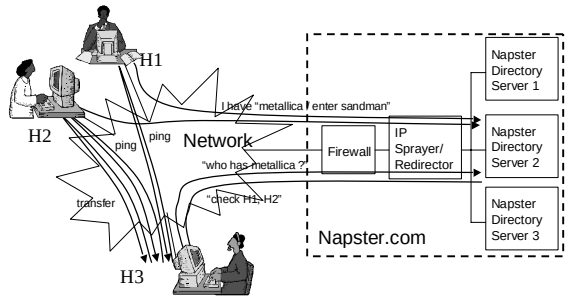
Napster Protocol



Napster Protocol



Napster Protocol



Napster Messages

General Packet Format

[chunksize] [chunkinfo] [data...]

CHUNKSIZE:
Intel-endian 16-bit integer
size of [data...] in bytes

CHUNKINFO: (hex)
Intel-endian 16-bit integer.

00 - login rejected	5B - whois query
02 - login requested	5C - whois result
03 - login accepted	5D - whois: user is offline!
0D - challenge? (nuprin1715)	69 - list all channels
2D - added to hotlist	6A - channel info
2E - browse error (user isn't online!)	90 - join channel
2F - user offline	91 - leave channel
	.....

From: http://david.weekly.org/code/napster.php3

Napster: Requesting a file

SENT to server (after logging in to server)
2A 00 CB 00 username
"C:\MP3\REM - Everybody Hurts.mp3"
RECEIVED
5D 00 CC 00 username
2965119704 (IP-address backward-form = A.B.C.D)
6699 (port)
"C:\MP3\REM - Everybody Hurts.mp3" (song)
(32-byte checksum)
(line speed)
[connect to A.B.C.D:6699]
RECEIVED from client
31 00 00 00 00 00
SENT to client
GET
RECEIVED from client
00 00 00 00 00 00
SENT to client
Myusername
"C:\MP3\REM - Everybody Hurts.mp3"
0 (port to connect to)
RECEIVED from client
(size in bytes)
SENT to server
00 00 DD 00 (give go-ahead thru server)
RECEIVED from client
[DATA]

Napster: Architecture Notes

- Centralized server:
  - single logical point of failure
  - can load balance among servers using DNS rotation
  - potential for congestion
  - Napster "in control" (freedom is an illusion)
- No security:
  - passwords in plain text
  - no authentication
  - no anonymity

Napster Issues

- Centralized file location directory
  - Single-level filesystem
  - Pose a bottleneck & vulnerability
- Need to partition to handle load
  - Strict partitioning based on client's IP address makes portion of the namespace invisible
  - Offering a unified view is computationally intensive, thus expensive – took more than a year for napster
- No replication, relies on keepalives to test client liveness
  - Also hard to scale, can cause packet storms, "train effect"

Napster Conclusions

- Technically not interesting
  - Centralized design, with bottlenecks
  - Simple implementation, 60-hour coding spree by company founder
- Immensely successful
  - Had 640000 users at any given moment in November 2000
- Success due to ability to create and foster an online community

Gnutella

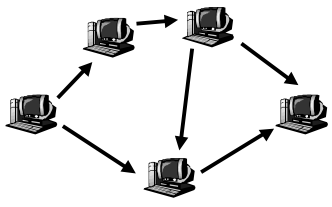
- peer-to-peer networking: applications connect to peer applications
- focus: decentralized method of searching for files
- each application instance serves to:
  - store selected files
  - route queries (file searches) from and to its neighboring peers
  - respond to queries (serve file) if file stored locally
- Gnutella history:
  - 3/14/00: release by AOL, almost immediately withdrawn
  - too late …
  - many iterations to fix poor initial design (poor design turned many people off)
- What we care about:
  - How much traffic does one query generate?
  - how many hosts can it support at once?
  - What is the latency associated with querying?
  - Is there a bottleneck?

Gnutella: How it works

Searching by flooding:

- If you don't have the file you want, query 7 of your partners.
- If they don't have it, they contact 7 of their partners, for a maximum hop count of 10.
- Requests are flooded, but there is no tree structure.
- No looping but packets may be received twice.

Flooding in Gnutella: loop prevention



Seen already list: "A"

Gnutella message format

- Message ID: 16 bytes (yes bytes)
- FunctionID: 1 byte indicating
  - 00 ping: used to probe gnutella network for hosts
  - 01 pong: used to reply to ping, return # files shared
  - 80 query: search string, and desired minimum bandwidth
  - 81: query hit: indicating matches to 80:query, my IP address/port, available bandwidth
- RemainingTTL: decremented at each peer to prevent TTL-scoped flooding
- HopsTaken: number of peer visited so far by this message
- DataLength: length of data field

Gnutella: initial problems and fixes

- Freeloading: WWW sites offering search/retrieval from Gnutella network without providing file sharing or query routing.
  - Block file-serving to browser-based non-file-sharing users
- Prematurely terminated downloads:
  - long download times over modems
  - modem users run gnutella peer only briefly (Napster problem also!) or any users becomes overloaded
  - fix: peer can reply "I have it, but I am busy. Try again later"
  - late 2000: only 10% of downloads succeed
  - 2001: more than 25% downloads successful (is this success or failure?)

Gnutella: Initial problems and Fixes (more)

www.limewire.com/index.jsp/net_improvements

- 2000: avg size of reachable network ony 400-800 hosts. Why so smalll?
  - modem users: not enough bandwidth to provide search routing capabilities: routing black holes
- Fix: Create peer hierarchy based on capabilities
  - Previously: all peers identical, most modem black holes
  - Connection preferencing:
    - favors routing to well-connected peers
    - favors reply to clients that themselves serve large number of files: prevent freeloading
  - Limewire gateway functions as Napster-like central server on behalf of other peers (for searching purposes)

Anonymous?

- Not anymore than it's scalable.
- The person you are getting the file from knows who you are. That's not anonymous.
- Other protocols exist where the owner of the files doesn't know the requester.

Gnutella Discussion

- What do you think?
- Good source for technical info/open questions: OK.

Problem

- P2P systems are used as scalable content distribution networks
- Main functionality provided: location of items based on key values
- No support for range queries!
  - Example: *find all objects with latitude between 16 and 18*

Goal

- Construct an index structure that supports
  - Equality and range queries
  - Peers insertion (join)
  - Peers deletion (leave)
- Space to store the index at each peer: sub-linear in the number of peers
- "Good" search/insertion/deletion performance

Some related work

- P2P environment:
  - Napster - use some centralized index
  - Gnutella - broadcast the query
  - Chord, Pastry, Tapestry, CAN - use hashing to construct indexes for efficient processing of equality queries
- Database community:
  - B+ trees

Model and assumptions

- No centralized control
- Peers
  - Own their data; expose the indexing attributes
  - Provide space and computation for the distributed index
- Query model
  - Equality queries: ex: object="tank" & load="high"
  - Range queries: ex: object="tank" & latitude<18 & latitude > 15
- Single numeric attribute index (val)
- No duplicates
- (One item per peer)

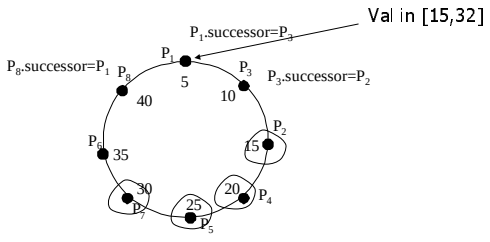
Talk Outline

- Motivating examples
- Processing high-speed data streams
- Peer-to-peer database systems
  - Introduction
  - P-tree data structure
  - P-tree search
  - P-tree maintenance
  - Experimental results

Ring structure

- Form a virtual ring based on the indexing attribute value stored at each peer (ex 5-10-20-5)
- For each peer, define:
  - *successor*: peer with the next value on the ring
  - *predecessor*: peer with the previous value on the ring
- Search: follow successor pointers - *O(N)* messages

Ring structure



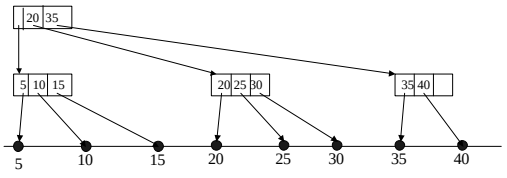
B+-trees

- Balanced search trees
- Each non-root level has between d and 2d (value,pointer) pairs
- Good update performance:
  - Higher levels are modified only if lower levels are
- Good search performance:
  - Separation
  - Coverage

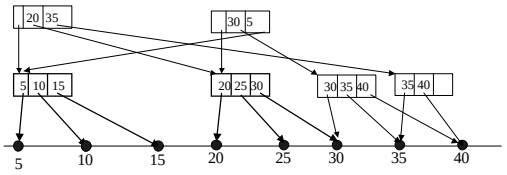
P(eer-to-peer)-trees

- Conceptually, each peer constructs a B+-tree with itself as the leftmost value
- Not feasible =>
  - Each peer constructs and stores only the nodes on the path from root to its leaf
  - Relies on other peers to complete the tree

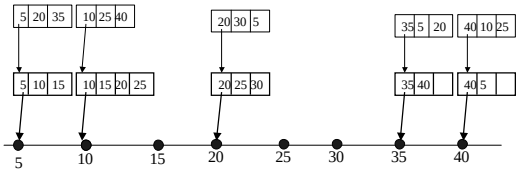
Example: B+ tree construction



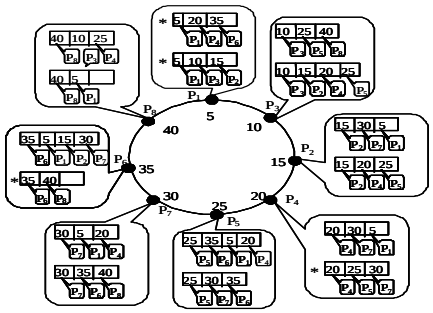
Example: B+ tree construction



Example: P-trees



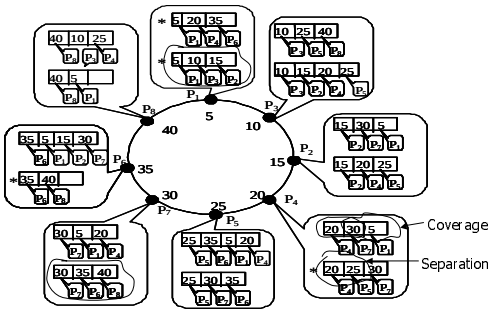
Example: P-trees, complete view



Invariants of P-trees

- If (key,ptr) and (key',ptr') are two consecutive entries in some P-tree at level i, then:
  - Separation: there are at least d entries at level i-1 in subtree rooted at ptr which are between key and key'
  - Coverage: there are no data values between the subtree rooted at ptr and the subtree rooted at ptr'
- Each node at a non-root level in the tree has at least d and at most 2d entries
- All the peers in the system can be reached

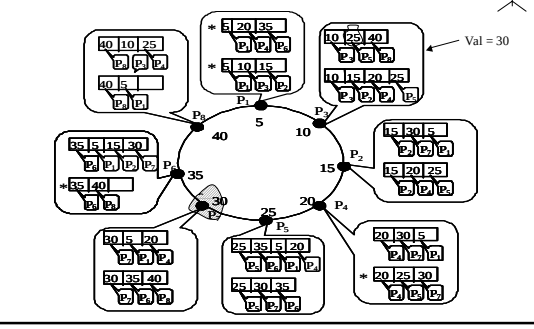
Example: P-trees, complete view



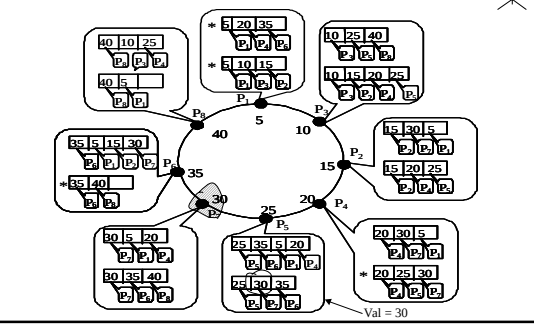
Talk Outline

- Motivating examples
- Processing high-speed data streams
- Peer-to-peer database systems
  - Introduction
  - P-tree data structure
  - P-tree search
  - P-tree maintenance
  - Experimental results

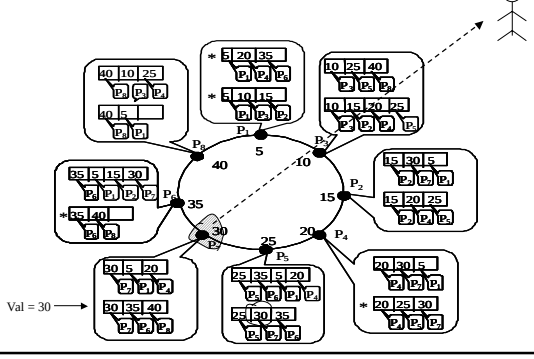
Search in P-trees



Search in P-trees



Search in P-trees



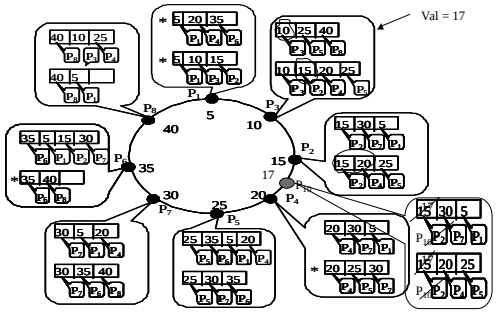
Talk Outline

- Motivating examples
- Processing high-speed data streams
- Peer-to-peer database systems
  - Introduction
  - P-tree data structure
  - P-tree search
  - P-tree maintenance
  - Experimental results

Insertion (peer joins)

- Assumption: each peer P that wants to join the system knows some other peer P' already in the system
- Steps to ensure consistency:
  - Add P to the virtual ring
    - Use Chord stabilization protocol
  - Initialize the P-tree of P
    - Copy the P-tree from the predecessor
  - Update the P-trees of existing peers to reflect the insertion of P
    - Rely on the Ping process and P-tree Stabilization process

Example: Insertion



Deletion (peer fails /leaves)

- Steps to ensure consistency:
  - Maintain virtual ring
    - Chord protocol maintains a list of successors
  - Update the P-trees
    - Rely on the Ping process and P-tree Stabilization process

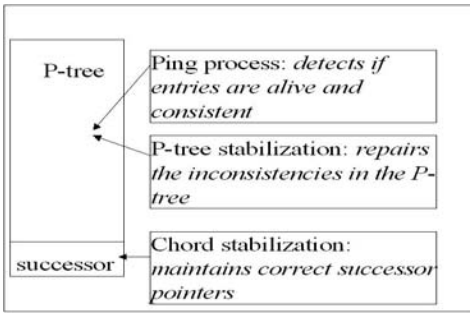
The Ping Process

- Runs periodically at each peer, starting with the lowest level in the P-tree
- Checks whether all entries in the P-tree nodes are "alive"
  - If not, the entry is removed from the node
- Check if all entries satisfy the Coverage and Separation property
  - If not, the entry is marked *inconsistent*
  - Note: we can use a "push-based" method where the children notify the parents about changes

The P-tree Stabilization process

- Runs periodically at each peer
- Fixes the inconsistencies detected by the Ping process
- The algorithm loops
  - from the lowest to top-most level
  - from left to right within each level
- If an entry is inconsistent, its sibling is replaced with a valid sibling (such that the P-tree invariants are maintained)

Maintenance algorithms



The End?

- CS330 versus CS432 versus CS530

- CS432 (Fall 2003)
- CS530 (Spring 2004)

What We Learned

- Database Systems
- The Middle Tier

What We Learned

- How to think about systems
- How to think about scalability
- How to think about systems design
- How to design algorithms

The End

A teacher is sombody who talks when everybody else sleeps.

Anonymous.